UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/650,257	08/27/2003	Bhavesh P. Davda	4366-120	7452

48500 7590 06/23/2010
SHERIDAN ROSS P.C.
1560 BROADWAY, SUITE 1200
DENVER, CO 80202

EXAMINER	
WEI, ZHENG	
ART UNIT	**PAPER NUMBER**
2192	
MAIL DATE	**DELIVERY MODE**
06/23/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte BHAVESH P. DAVDA

Appeal 2009-005198
Application 10/650,257
Technology Center 2100

Decided: June 23, 2010

Before HOWARD B. BLANKENSHIP, JAY P. LUCAS, and
THU A. DANG, *Administrative Patent Judges*.

BLANKENSHIP, *Administrative Patent Judge*.

DECISION ON APPEAL

STATEMENT OF THE CASE

This is an appeal under 35 U.S.C. § 134(a) from the Examiner's final rejection of claims 1-21, which are all of the claims in this application. We have jurisdiction under 35 U.S.C. § 6(b).

We affirm-in-part.

Invention

Appellant's invention relates to a method and apparatus for updating running processes. A jump instruction is injected into the first instruction line of a function that has been updated. The jump instruction redirects the program to a location within a jump table containing the address of the first instruction of an updated function. Injection of the jump instruction can be made without stopping execution of the application, thereby allowing a patch to be installed without interrupting application services. Abstract.

Representative Claims

1. A method for updating a running process, comprising:

allocating in executable program code text first memory space operable to receive new program instructions comprising at least a first updated function;

allocating in executable program code text second memory space operable to receive address information related to said new program instructions;

running said executable program code;

stopping execution of said executable program code;

injecting a jump instruction and an address of an update table at a location in a memory containing a first instruction of a first replaced function, wherein said address of said update table contains an address of a first instruction of said first updated function; and

resuming execution of said executable program code, wherein said first updated function is called in place of said first replaced

function, and wherein said executable code is updated in said memory.

6. The method of Claim 1, further comprising:

determining a first distance between a position within said code text at which execution of said executable program code is stopped and an address of a first function, wherein said first function is a function to be updated; and

in response to said first distance exceeding a predetermined amount, populating an update table stored in memory with an address of a first updated function.

Examiner's Rejections

Claims 1-5, 7, 10-13, and 16-19 stand rejected under 35 U.S.C. § 102(b) as being anticipated by Lillich (US 5,619,698).

Claims 6, 8, 14, and 15 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Lillich and Buban (US 2004/0107416 A1).

Claim 9 stands rejected under 35 U.S.C. § 103(a) as being unpatentable over Lillich and Munsil (US 2003/0167463 A1).

Claims 20 and 21 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Hundt (US 2002/0152455 A1) and Buban.

Claim Groupings

In view of Appellant's arguments in the Appeal Brief, we will decide the appeal on the basis of claims 1 and 6. *See* 37 C.F.R. § 41.37(c)(1)(vii).

ISSUES

(1) Has Appellant shown that the Examiner erred in finding that Lillich describes "injecting a jump instruction" in the manner required by claim 1?

(2) Has Appellant shown that the Examiner erred in finding that the combination of Lillich and Buban teaches determining a first distance between a position within code text at which execution is stopped and an address of a first function to be updated, as required by claim 6?

FINDINGS OF FACT

Lillich

1. Lillich describes a well known 68K patching paradigm 100 (Fig. 2). The system routines for the 68K operating system reside mainly in ROM. However, to provide flexibility for any subsequent development, application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM. This indirect mechanism permits the ROM addressing of system routines to vary, or to be replaced by patch routines, without affecting the operation of applications which utilize the system routines. Col. 2, ll. 1-23.

2. The indirect calling mechanism of the 68K patching paradigm 100 is further implemented by utilizing a feature of the 68K series microprocessor called a trap. Col. 2, ll. 34-47.

3. During execution of application code 102, the 68K microprocessor will encounter and execute the ATRAP 104. In response, the microprocessor pushes a portion of the machine state onto the computer

system's stack, the state including the address of the ATRAP instruction 104. The microprocessor resumes execution at the address indicated in a low memory location, which is the address of the trap dispatcher 108. Once executing, the trap dispatcher 108 examines the bit pattern of the ATRAP instruction 104 to determine what operation it stands for, looks up the address of the corresponding system routine in the trap table 110, and then jumps to the corresponding system routine. Col. 2, l. 48 to col. 3, l. 5.

4. Because the trap table 110 is resident in RAM 130, individual entries in the trap table 110 can be changed to point to addresses other than the original ROM addresses. This allows the system routines to be replaced or augmented by patches. At startup time the system can load new versions of individual routines (e.g. from the System file or from a floppy disk) into RAM and then patch the trap table in order to redirect any calls to the original system routine to the new versions. Additionally, when new applications are launched they too can load new versions of individual routines into RAM and then patch the trap table in order to redirect any calls to the original system routine to the new versions. Col. 3, ll. 26-38.

5. One example of a redirection to a patched system routine is symbolized in Figure 2 by dashed flow control lines 150 and 152. An ATRAP instruction 104 calling a patched system routine will initiate a process in which the trap dispatcher 108 will look up the system routine corresponding to the ATRAP instruction 104. However, in the patched case, address_1 will point to patch code 132 located in RAM 130 rather than the original system routine. Thus dashed flow control line 150 illustrates jumping to the patch code 132 and dashed flow control line 152 illustrates

jumping back to the application code 102 after the patch code 132 has finished executing. Col. 3, ll. 39-50.

Buban

6. Buban teaches that patching code during execution may be difficult because of interrupts, multiple threads running on multiple processors, multi-thread switching between the patch-related code and the running code on a single processor. To safely patch running code, an instruction (existing_instr2) is replaced by a jump command (jmp patch). The replaced instruction (existing_instr2) may need to be completely contained within a processor's smallest unit of atomically replaceable memory, (e.g., a 64-bit (8-byte) word on an Intel x86 processor), so that no processor that might execute the subroutine will see a partially updated version of the patched routine. Preventing execution of a subroutine that is being patched simultaneous (in time) to applying the patch eliminates this requirement. ¶ [0046].

7. To find a suitable instruction to replace, it may be necessary to move backwards in the existing code, e.g., into existing_instr1, and then add any such tail instructions to the beginning of the patched code, or to move forward, e.g., into existing_instr3 and then add instructions to the front of the patched instructions (e.g., patched_instructionA) to undo the instructions that were executed but should have been patched over. Although inefficient, code that is likely to need a hotpatch in the future may initially be programmed with future hotpatching in mind, and have special no-op instructions or meaningless jump or other instructions intentionally placed into it for this future purpose. ¶ [0047].

PRINCIPLES OF LAW

Claim Interpretation

During examination, claims are to be given their broadest reasonable interpretation consistent with the specification, and the language should be read in light of the specification as it would be interpreted by one of ordinary skill in the art. *In re Amer. Acad. of Sci. Tech Ctr.*, 367 F.3d 1359, 1364 (Fed. Cir. 2004) (citations omitted). The Office must apply the broadest reasonable meaning to the claim language, taking into account any definitions presented in the specification. *Id.* (citations omitted).

Unless the steps of a method actually recite an order, the steps are not ordinarily construed to require one. *Interactive Gift Express, Inc. v. CompuServe, Inc.*, 256 F.3d 1323, 1342 (Fed. Cir. 2001). However, such a result can ensue when the method steps implicitly require that they be performed in the order written. *Id.* (citations omitted). *See also Altiris, Inc. v. Symantec Corp.*, 318 F.3d 1363, 1369-71 (Fed. Cir. 2003) (district court erred in claim construction by reading a step order from the written description into the claims).

Anticipation

"Anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, arranged as in the claim." *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 730 F.2d 1452, 1458 (Fed. Cir. 1984).

Obviousness

"What matters is the objective reach of the claim. If the claim extends to what is obvious, it is invalid under § 103." *KSR Int'l Co. v. Teleflex, Inc.,* 550 U.S. 398, 419 (2007). "The combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." *Id.* at 416.

ANALYSIS

Claims 1-5, 7, 9-13, and 16-19

Appellant contends that Lillich makes use of a pre-existing trap instruction in the application code. Therefore, according to Appellant, Lillich does not inject an instruction in memory containing a first instruction of a function to be replaced as required by claim 1. Br. 10-11.

Appellant's contention appears based on the premise that claim 1 requires that "injecting" occurs between "stopping" and "resuming." However, this order is not required by the claim or the Specification. For example, "the patching or substitution of the original function 208 with an updated function 216 can be performed while the executable 204 program is running" (Spec. 10:18-19), which implies that the patching, which includes "injecting" (Spec. 8:11-13), can also be performed while the executable program is not running. Therefore, "injecting" as recited in claim 1, when read in light of the Specification, can occur when the trap instruction is written into the code.

Lillich describes a trap instruction that causes a microprocessor to resume execution of code at an address of a trap dispatcher. The trap dispatcher examines the trap instruction, looks up an address of a

corresponding routine in a trap table, and jumps to the corresponding system routine. Because the trap table is resident in RAM, individual entries in the trap table can be changed to point to addresses of patch routines. FF 1-5. Therefore, Lillich describes "injecting a jump instruction and an address of an update table at a location in a memory containing a first instruction of a first replaced function, wherein said address of said update table contains an address of a first instruction of said first updated function" within the meaning of claim 1.

Lillich further discloses that when the microprocessor encounters a trap instruction, the microprocessor resumes execution of the code at the address of the trap dispatcher (FF 3), which indicates that the microprocessor stops execution when encountering the trap instruction. Therefore, Lillich describes "stopping execution" and "resuming execution" within the meaning of claim 1.

We therefore sustain the § 102(b) rejection of claim 1. Appellant has not provided arguments for separate patentability of dependent claims 2-5 and 7. Because we find the arguments for claim 1 unpersuasive, we sustain the § 102(b) rejection of claims 2-5 and 7.

Appellants also have not provided arguments for separate patentability of claims 9-13 and 16. Although Appellant places claims 18 and 19 in a separate heading in the Appeal Brief, the arguments[1] rely on the same presumption that the claims require stopping execution of code, performing a patch operation, and then resuming execution of program code. However,

---

[1] "A statement which merely points out what a claim recites will not be considered an argument for separate patentability of the claim." 37 C.F.R. § 41.37(c)(1)(vii).

Appellant has not shown that the invention "as claimed" requires the steps in the order alleged. We therefore sustain the § 102(b) rejection of claims 10-13 and 16-19 and the § 103(a) rejection of claim 9.

Claims 6, 8, 14, and 15

The Examiner finds that paragraphs [0046] and [0047] of Buban teach determining a distance between two memory addresses. In particular, the Examiner finds that Buban teaches moving backwards in existing code and adding instructions to the beginning of patched code, or moving forward and adding instructions to the front of the patched instructions. The Examiner concludes that in order to add instructions, the process taught by Buban would have to determine distance between codes; otherwise, it would be inoperative. Ans. 9, 14-15.

However, moving to the beginning of patched code, or to the front of patched instructions, does not, by itself, teach determining a distance. The rejection fails to show how moving backwards to the beginning of patched code, or moving forward to the front of patched instructions, teaches determining a first distance between a position within said code text at which execution of said executable program code is stopped and an address of a first function.

Claim 14 contains a limitation similar to that of claim 6 for which the rejection fails. Claim 8 depends from claim 6, and claim 15 depends from claim 14. We thus do not sustain the § 103(a) rejection of claims 6, 8, 14, and 15.

Claims 20 and 21

Claim 20 recites "a signal handler tool operable to replace in memory an address of said function to be replaced with an address of a replacement function in response to said position of said instruction pointer being at least a predetermined distance from said address of said replacement function." The Examiner relies on paragraph [0047] of Buban to show this limitation. Ans. 15-16. However, as discussed in the analysis of claim 6, the rejection fails to show how paragraph [0047] of Buban teaches determining a distance. Therefore, the rejection fails to show how Buban teaches replacing an address of a function with an address of a replacement function "in response to said position of said instruction pointer being at least a predetermined distance from said address of said replacement function" as recited in claim 20.

Claim 21 depends from claim 20. We thus do not sustain the § 103(a) rejection of claims 20 and 21.

CONCLUSIONS OF LAW

(1) Appellant has not shown that the Examiner erred in finding that Lillich describes "injecting a jump instruction" consistent with the requirements of claim 1.

(2) Appellant has shown that the Examiner erred in finding that the combination of Lillich and Buban teaches determining a first distance between a position within code text at which execution is stopped and an address of a first function to be updated, as required by claim 6.

DECISION

The rejection of claims 1-5, 7, 10-13, and 16-19 under 35 U.S.C. § 102(b) as being anticipated by Lillich is affirmed.

The rejection of claims 6, 8, 14, and 15 under 35 U.S.C. § 103(a) as being unpatentable over Lillich and Buban is reversed.

The rejection of claim 9 under 35 U.S.C. § 103(a) as being unpatentable over Lillich and Munsil is affirmed.

The rejection of claims 20 and 21 under 35 U.S.C. § 103(a) as being unpatentable over Hundt and Buban is reversed.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a). *See* 37 C.F.R. § 41.50(f).

AFFIRMED-IN-PART

msc

SHERIDAN ROSS P.C.
1560 BROADWAY, SUITE 1200
DENVER CO 80202